UCRL-MA-140943

ParaDyn A Parallel Nonlinear, Explicit, Three-Dimensional Finite-Element Code for Solid and Structural Mechanics User Manual

C. G. Hoover, A. J. De Groot, R. J. Sherwood

June 1, 2000

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

DISCLAIMER

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

ParaDyn

A Parallel Nonlinear, Explicit, Three-Dimensional Finite-Element Code for Solid and Structural Mechanics

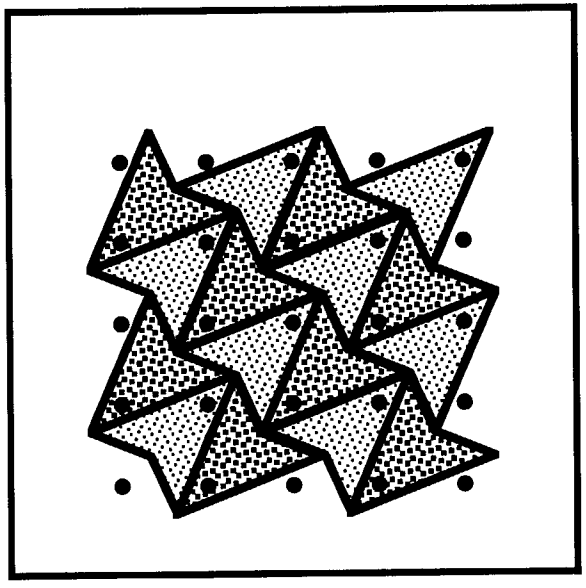
User Manual

Carol G. Hoover, Anthony J. De Groot, Robert J. Sherwood

Methods Development Group
Mechanical Engineering

ParaDyn Version 1.01
June 2000

Butterfly Instability



Preface

This User Manual documents the collaborative code development work with my colleagues, Tony De Groot and Bob Sherwood. Our efforts represent the majority of the original work in the parallel algorithm design and development for the ParaDyn Project. Doug Speck, Elsie Pierce, and Vic Castillo are now substantive contributors to the ParaDyn Project. Doug and Elsie, in particular, have made it possible to quickly design flexible binary databases (MILI) and have developed significant enhancements to the GRIZ visualization software. Their work paves the way for the visualization capabilities needed when using parallel computers with hundreds and thousands of processors.

Analysts provide the insights needed to turn our software development into an effective production tool for finite-element engineering analysis. At the Lawrence Livermore National Laboratory (LLNL) Dan Badders, Tony Lee, and Tony DePiero entered the turbulent waters of massively parallel computers earliest and have provided very valued input to our code development efforts.

Raju Namburu and Photios Papados are collaborators from the Army Research Laboratory (ARL) and the Engineer Research and Development Center (ERDC). They led the way to the first multimillion element calculations on Department of Defense high performance computers. John Benner and colleagues from the Los Alamos National Laboratory (LANL) were the first to use the ParaDyn program on over 1000 processors.

We especially appreciate the very valuable comments and support that our collaborators from ARL, ERDC, and LANL have provided for our code development efforts.

Carol Hoover
Methods Development Group
Mechanical Engineering Department
Lawrence Livermore National Laboratory
Livermore, California 94551-7808
925-422-1556 hoover1@llnl.gov

Abstract

ParaDyn is a parallel version of the DYNA3D computer program, a three-dimensional explicit finite-element program for analyzing the dynamic response of solids and structures. The ParaDyn program has been used as a production tool for over three years for analyzing problems which range in size from a few tens of thousands of elements to between one-million and ten-million elements. ParaDyn runs on parallel computers provided by the Department of Energy Accelerated Strategic Computing Initiative (ASCI) and the Department of Defense High Performance Computing and Modernization Program. Preprocessing and post-processing software utilities and tools are designed to facilitate the generation of partitioned domains for processors on a massively parallel computer and the visualization of both resultant data and boundary data generated in a parallel simulation. This manual provides a brief overview of the parallel implementation; describes techniques for running the ParaDyn program, tools and utilities; and provides examples of parallel simulations.

TABLE OF CONTENTS

1.0 BACKGROUND

Significant speed gains (for example, factors of sixty or more on sixty-four processors) are being achieved for engineering design calculations on parallel computers with as few as a hundred processors using ParaDyn, the parallel version of the DYNA3D program [1]. The latest massively parallel computers with thousands of processors now make it possible to design engineering models for mechanical system analysis with multiple component parts as well as to add more detail and complexity in the models for components. ParaDyn and DYNA3D are explicit finite-element programs designed to solve for the nonlinear, transient response of solids and structures. The two programs, contained within a single source, are developed by the Methods Development Group at the Lawrence Livermore National Laboratory (LLNL). The web site, http://www.llnl.gov/eng/mdg/mdg_home.html, provides general information and online documentation about the complete family of themomechanical programs developed in the Methods Development Group.

Parallel computers are now commonplace in our computing environment. Computers with speeds ranging from 1 to 100 TeraOps (10^{12} to 10^{14} operations per second) and thousands of processors will be delivered through the year 2004 to the Accelerated Strategic Computing Initiative (ASCI) Program. These computers dominate the high-end computing at LLNL. More modest parallel workstation clusters with ten or twelve processors support the Institutional Programs at LLNL. ParaDyn is implemented using the Message Passing Interface (MPI) standard. This standard makes it possible to develop parallel engineering applications that run on both distributed memory massively parallel computers and also shared memory workstation clusters. A summary of the computer resources at LLNL may be viewed at http://www.llnl.gov/asci/platforms and http://www.llnl.gov/sccd/FAST.resources.html.

The Department of Defense High Performance Computing and Modernization Program is also a very strong participant in acquiring hardware and developing software for next-generation massively parallel computers. Computer characteristics in terms of size, speed, and number of processors at the DOD Major Shared Resource Centers (MSRCs) are summarized in the links provided by the Program Office web site at http://www.hpcmo.hpc.mil. These data illustrate the continuing significant increase in our parallel computational speeds and capacities.

ParaDyn is a parallel production program. Code development efforts are now directed toward optimizing the parallel algorithms, developing parallel preprocessing and post-processing software, and developing software tools for engineering optimization studies. Concurrent with the

parallel development effort, many code developers for the DYNA3D program are contributing new algorithms, elements, and material models to enhance the mechanics modeling capabilities. A single source encompassing both the ParaDyn and DYNA3D algorithms makes it possible to naturally migrate the DYNA3D enhancements into the production ParaDyn program. Finally, coupled programs for thermal, mechanical, and fluid analysis are being designed for parallel computers. A production capability for coupled analysis is possible in the next few years.

2.0 OVERVIEW OF PARADYN

2.1 Introduction

Advances in the development of parallel algorithms for explicit finite-element analysis and domain partitioning techniques have led to scalable production applications using ParaDyn. This has resulted in several benefits to our engineering design programs. Firstly, calculations are now performed in a day or less for problems that previously ran over several weeks. Secondly, new models are being generated for mesh sizes between one-million and ten-million elements. This is an order of magnitude larger than the largest models possible in the past. Finally, longer time simulations (problems running for a few million steps) for small to moderate sized problems are now being run on both massively parallel computers and workstation clusters.

Analysts play an important new role in preparing models for parallel computers. The meshes are increasingly much larger and more complex. New validation tools are needed for the mesh generation step, specification of boundary loads and constraints, and defining facets on interfaces. In addition, the modeling of contact interfaces can have a significant effect on the parallel performance and scalability. Optimizing the performance and achieving scalability of parallel contact algorithms is particularly challenging. Two distinct forms of parallel contact algorithms have been developed in ParaDyn and will be discussed in the sections on parallel contact.

Current and future parallel algorithm development is focused on contact algorithm research and also on providing the necessary automated modeling tools for the analyst. The development of the GRIZ4 visualization tool [2] based on the new MILI (Mesh I/O Library) software [3] is one step toward achieving this goal. In addition, we anticipate further research in numerical methods to enhance the scalability of parallel contact algorithms.

This manual is available in PDF form for interactive viewing. In some sections the text is displayed in either red or blue. The blue highlighting indicates text passages that are included in a version of the manual designed for a specific computer or site location. Thus, the blue text indicates conditional text which has been generated with the particular version of the manual being viewed. Red text is used to highlight cautionary notes. For instance, red text is used to highlight a warning message to copy output files between ParaDyn runs when those files would otherwise be overwritten by a subsequent run in the same directory.

To prepare input data and select control options and flags for the ParaDyn program, follow the discussions in the DYNA3D User Manual [1]. An online manual for DYNA3D is available on the home page for the Methods Development Group, http://www.llnl.gov/eng/mdg/mdg_home.html. Instructions for using the new preprocessing and post-processing software for parallel simulations are included in this manual. In this manual the term *preprocessing* refers to the software used after the mesh generation step to prepare the model for the analysis with the ParaDyn program. More specifically it refers to the partitioning of the model. Computer-specific instructions and example problems are provided here to illustrate how to run a ParaDyn analysis on a parallel computer. Further documentation for the preprocessing software for partitioning meshes for ParaDyn is available online with the ParaDyn software. This documentation plus installation instructions are also available in an LLNL UCRL Report [4]. And finally, some standard features in DYNA3D requiring additional documentation for a parallel analysis are included in this manual.

A set of typeface conventions is followed throughout this manual to allow the reader to easily distinguish between **commands**, *parameters*, and `computer generated text`. **Commands** that appear in **bold** type should be entered verbatim. *Parameters* that appear in *italic* type should be given values when included in the input. `Computer generated text`, such as error messages or default file names, is printed in a `typewriter-like (Courier)` font. In text passages file names appear in italic type for clarity.

The next sections provide introductory discussions about parallel algorithms and computers. Section 2.2 discusses the parallel finite-element model and describes partitioning methods. Section 2.3 discusses parallel performance and scalability measurements. Section 2.4 characterizes contact interfaces and their implementation in parallel. This section concludes with guidelines for modeling contact interfaces with parallel algorithms.

2.2 The Parallel Finite-Element Model

A successful strategy for parallel implementation of the explicit finite-element method is based on dividing the mesh among the processors and executing ParaDyn on a subdomain in each processor [5]. The elements from the mesh are divided into subdomains so that each processor has approximately the same amount of calculations to perform in a timestep. The nodes on the boundaries of a subdomain are referred to as shared nodes. Nodal force data for shared nodes are communicated between processors when the nodal force updates are calculated. The nodal points

on the subdomain boundaries may be duplicated (shared) in more than one processor. Mesh partitioning is the strategy for dividing the problem into subdomains and mapping subdomains to processors. This is illustrated for two processors in Figure 1.

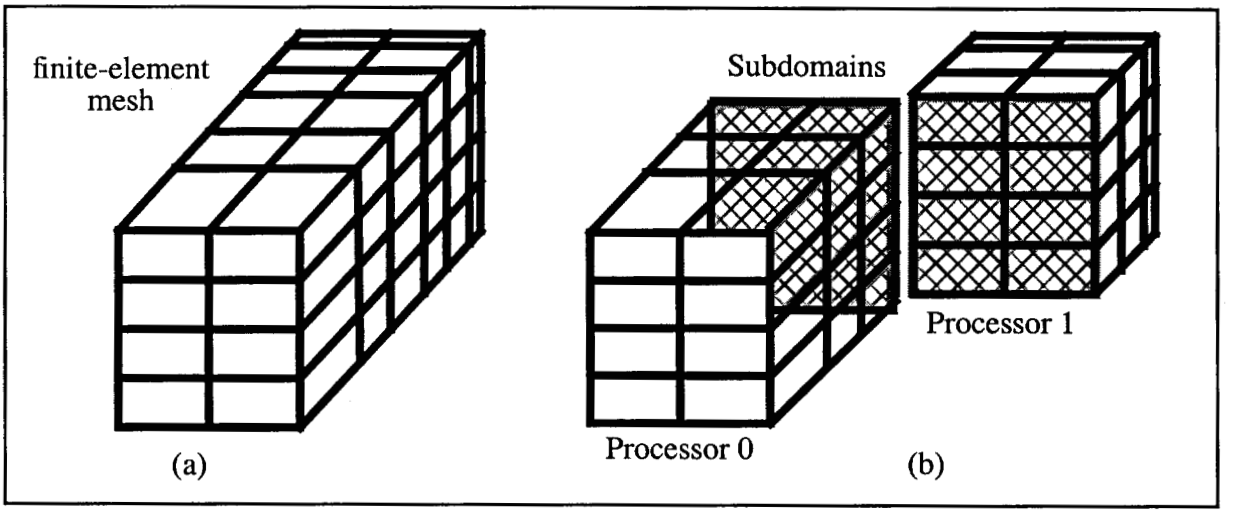


Figure 1. Two subdomains on a finite-element mesh. (a) The original mesh with 48 elements is partitioned into two subdomains with 24 elements each. (b) The calculations involving nodal points on the cut plane (shown as patterned) are performed in both processors. The 15 nodes on the cut plane are referred to as shared nodes.

The nodal forces consist of contributions from applied loads, contact interactions, and internal deformations.

$$\mathbf{F}_{node} = \mathbf{F}_{applied} + \mathbf{F}_{contact} - \mathbf{F}_{internal}$$

The internal force calculation for a shared node includes a contribution from elements in different processors. Each processor calculates a partial nodal force for the elements in its processor. These partial force contributions are communicated between the processors so that the total force computed for a shared node is the same in all processors within the error introduced by the ordering of the calculations. As much as possible, the ParaDyn algorithms are designed to store partial nodal force data for shared nodes until all contributions to the nodal force have been computed before communicating the shared data.

Research in applied mathematics has led to efficient techniques for subdividing or partitioning the complicated unstructured meshes that arise in practical engineering applications [6-9]. We use the METIS software from the University of Minnesota to partition finite-element meshes and contact

surfaces. (For more information on METIS, see http://www-users.cs.umn.edu/~karypis/metis/main.shtml). The METIS algorithms use a graph to represent the finite-element mesh. Preprocessing software automatically produces the graphs needed for the mesh partitioning step.

Mesh partitioning is accomplished by representing a finite-element mesh as a graph. A graph has vertices and interconnecting edges. The vertices and edges represent objects on the mesh. For finite-element meshes, the vertices of the graph correspond to elements (zones) in the mesh and the edges correspond to nodes in common between two connected elements. This is illustrated in Figure 2.

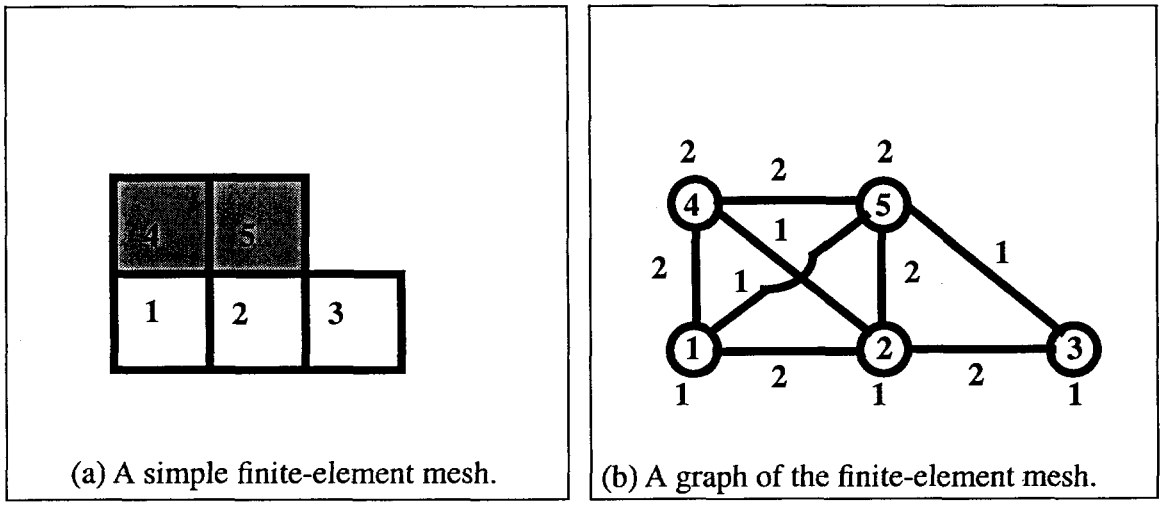


Figure 2. A finite-element mesh and the graph representation of the mesh. (a) This simple mesh consists of two materials, shaded and unshaded. The shaded material requires twice as much calculation time as the unshaded material. (b) This is the graph of the mesh. The vertices are represented as circles and the number near a vertex is the computational weight of the vertex. The lines connecting the vertices are edges and represent the shared data between the vertices. The number specified along the edges represents the number of shared nodes between the two elements represented by the vertices.

The graph represents the element-to-element connectivity for the mesh. The METIS algorithms find an optimal division of the graph corresponding to a specified number of subdomains. An important aspect of graph partitioning techniques is the use of weighting factors for both the vertices and edges. These weighting factors are used to balance the vertices into sets of roughly equal weight and thus, provide intelligent input control over the partitioning of the mesh. To illustrate this, the relative computational cost for a complex material model, a boundary condition or any other expensive part of the calculation, can be used to weight the vertices. Similarly, an edge

in the graph represents the number of common nodes between the elements and can be appropriately weighted by a relative measure of the shared data communicated if the graph is cut on that edge. Figure 2b indicates the edge weights used for the few elements shown in Figure 2a.

The partitioning task is automated completely in the preprocessing tool, DYNAPART, for any mesh geometry. This software was used for the assignment of a one-million element mesh to 128 processors as shown in Figure 3. The colors are used to show the processor assignment for subdomains on the mesh. The mesh was developed to model a shock moving from the top vertex of the mesh in the direction of the half-cylindrical cavity region located half way down and on the left-hand side of the mesh. The mesh is zoned very finely at the top of the model to resolve the shock structure. As a result of this fine zoning the subdomains are much smaller at the top of the mesh than the subdomains on the lower part of the mesh where the zoning is coarse. The mesh lines are not shown in the Figure.

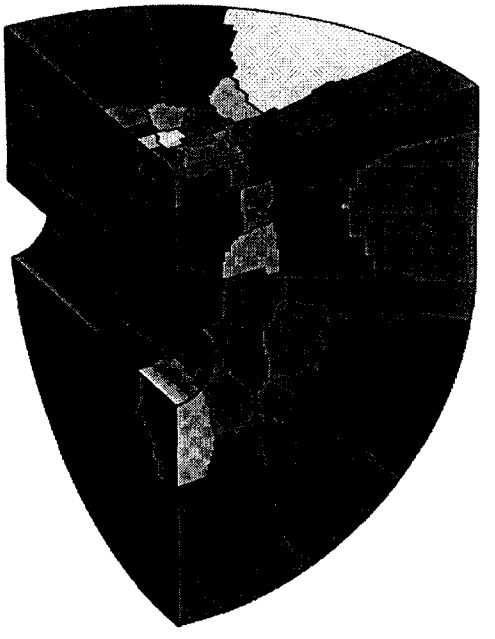


Figure 3. Processor assignment for a one-million element mesh. Colors are used to distinguish the subdomains assigned to 128 processors. This problem without sliding interfaces scales linearly as the number of processors is increased up to roughly 1000 processors.

2.3 Parallel Performance and Scalability

The time required to deliver results on a parallel computer is the sum of the time for computing results on the processors and the time for communicating data between the processors.

$$\tau_{wc} = \tau_{calc} + \tau_{comm}$$

where

τ_{wc} is the total elapsed time taken for the calculation (total wall clock time);
τ_{calc} is the elapsed time the processors spend computing results;
τ_{comm} is the elapsed time the processors spend communicating shared data.

Ideally, if the number of processors is doubled, the rate for delivering results will be doubled. In practice this linear scaling of the delivery rate with the number of processors breaks down when the communication time becomes significant compared to the amount of time processors spend computing results.

If the parallel calculation is efficient, the delivery time τ_{wc}, for N_p processors will be approximately equal to $1/N_p$ of the time for calculating the same results on one processor, τ_1. Thus, a measure of the parallel efficiency is given by the following:

$$\varepsilon = \tau_1/(N_p \tau_{wc}) \times 100\%.$$

The preprocessing software tools in DYNAPART provide convenient methods for selecting the maximum optimal number of processors to use for a ParaDyn simulation. Using more processors will result in no further improvement in delivering results and wastes computing resources.

The optimal number of processors to select for a parallel calculation *without contact* can be estimated by specifying the minimum number of elements that can be allocated to a processor, which results in a negligible amount of communication time (ideally less than 10%) compared to the computation time for the deformation calculations. The statistics calculated by the partitioning software provide quantitative values for determining the optimal number of processors. These statistics will be discussed in the section describing the partitioning software tools. For current models, an estimate for the minimum number of elements to assign to a processor is between 1000 and 2000.

Almost all problems of interest include contact interface definitions, which are always more difficult to run efficiently in parallel.

2.4 Scalable Parallel Contact Algorithms

Designing efficient parallel contact algorithms is challenging because the surface motion is often unpredictable and new interfaces may appear dynamically. As a result, dividing the problem domain into subdomains that are optimal for calculating the element deformations will almost never result in an efficient division of the problem for the contact calculations. Hence, the ParaDyn software uses partitioning methods for the sliding interfaces that differ from the partitioning for the mesh. The results for the contact force calculations are communicated once in a timestep to the processors defined by the mesh partition. Similarly the nodal position and velocity updates are communicated from the processors defined by the mesh partition to the processors defined by the contact partitions once in a timestep.

There are currently two parallel contact algorithms in the ParaDyn program [10] and selecting the algorithm to use will depend on the predictability of the surface motion as well as the relative size of the largest interfaces. In some problems, surfaces initially close together engage in small relative motion and the contact remains in a localized region of the mesh. We refer to this as local contact. Sliding interface types (1-3, 5-10) in DYNA3D/ParaDyn are local algorithms. For other problems with large deformation or moving parts, the motion of the surfaces is not predictable. Thus, more sophisticated and computationally expensive searches for the surfaces in contact must be performed throughout the simulation. We refer to this as arbitrary contact. Examples illustrating arbitrary contact are a ball rolling on a plane, a surface folding on itself, or an automobile crash simulation. Arbitrary contact is implemented in DYNA3D automatic contact interfaces, types 12 through 14. The automatic contact algorithm type 14 is the material erosion algorithm (SAND).

The parallel algorithm implementing local contact allocates a contact interface (both the master surface and the slave segments or nodes) into one processor. This method is very efficient and useful for problems with many contact surfaces that are relatively small. The method can limit the scalability of the problem if there are large contact surfaces because the partitioning for contact may require more elements in a processor than is efficient for an optimal mesh partitioning. In this latter case, the problem can often be made more scalable by using the parallel automatic contact algorithms described below.

The second parallel contact algorithm models arbitrary interface contact. The search for arbitrary contact is implemented in the DYNA3D automatic contact algorithms. The method uses a set of cells (buckets) for sorting and grouping surface nodes and segments into localized regions on the mesh. The parallel version of the algorithm partitions the buckets among the processors to balance

the contact calculations among them and minimize the communication of nodes and surface faces in cells with data that must be shared between processors. An additional step is needed in the automatic contact algorithm when contact surface motion requires a regeneration of the cells and sorting of the surfaces into the new cells. The frequency of the bucket-regeneration step is automatically computed or, in some unusual situations, can be specified as a user input. In ParaDyn the bucket-regeneration step induces extra communication during the timestep over which it occurs. Thus, it is important to rebucket as infrequently as possible to avoid the extra communication costs.

The arbitrary contact algorithms are referred to as automatic contact algorithms in the DYNA3D documentation. The reason for this is that the faces on the contact surfaces are generated automatically in DYNA3D and ParaDyn. This is in contrast to the older more tedious method in which the contact interfaces are defined by the analyst when generating the mesh. For the large meshes generated for parallel simulations this is an extremely time consuming task. Furthermore, the contact interface definition is often a source of errors in the input file because of difficulties encountered during the part-merging phase of the mesh generation.

The parallel contact algorithm in ParaDyn is selected by the DYNA3D sliding interface method used for modeling the contact. Sliding-interface types (1-3, 5-10) are implemented by assigning each contact interface into a processor. Sliding interface type 11 (SAND) is not implemented in parallel because the equivalent interface is modeled in the more robust and newer automatic contact interface type 14. The single surface contact algorithm in type 4 is implemented as a full surface assigned to a processor. A more general version of single surface contact is contained within interface types 12 or 13.

Sliding interface types 12 and 13 are identical in their parallel implementation. However, type 13 provides options for controlling the search and other features as follows:

Boxes can be defined to limit the domain of the search;
Material can be included or excluded in the boxes for the search;
Faces defined as in a type 3 interface can be specified rather than automatically generated.

See the DYNA3D user manual in the keyword options section for a description of the input for sliding interface type 13.

The most commonly used sliding interface in DYNA3D is type 3. The same physical interface condition (sliding with friction and voids) can also be modeled with the automatic contact algorithm types 12 and13. The parallel efficiency for either choice for the sliding interface may vary significantly depending on the details of the interfaces and the size of the model. The next sections compare the two forms of parallel contact algorithms. These comments provide guidelines for selecting which algorithm to use for an efficient parallel calculation.

2.4.1 Parallel Local Contact

The parallel local algorithm allocates each sliding interface fully in one processor. For more than one sliding interface, the DYNAPART preprocessing software distributes the set of interfaces among as many processors as possible. Special graph weighting methods are used for partitioning the contact interfaces and evenly distributing the contact calculations among processors.
This algorithm is illustrated with the spin forming mesh shown in Figure 4. This model consists of a rotating plate formed into a hemispherical shape by rollers in contact with the plate surface. A four-processor partition for this problem cuts the plate into three concentric rings. The center ring and brushes form the sliding interface and are fully contained in one processor. An eight-processor partition for the problem cuts two of the rings in half. The sliding interface, the center ring and the rollers, remains uncut.

The number of processors used for problems with local contact is often limited by the largest sliding interface. If the surfaces on the largest interface contain many more elements than the minimum number of elements that will provide a balanced calculation for the element deformation, then the optimal number of processors for the calculation will be determined by the contact calculations. A more subtle condition for large contact surfaces occurs when the partitioning cuts the mesh one element below the surfaces in the sliding interface. This results in a large amount of shared-node communication for processors with elements connected to the contact surfaces.

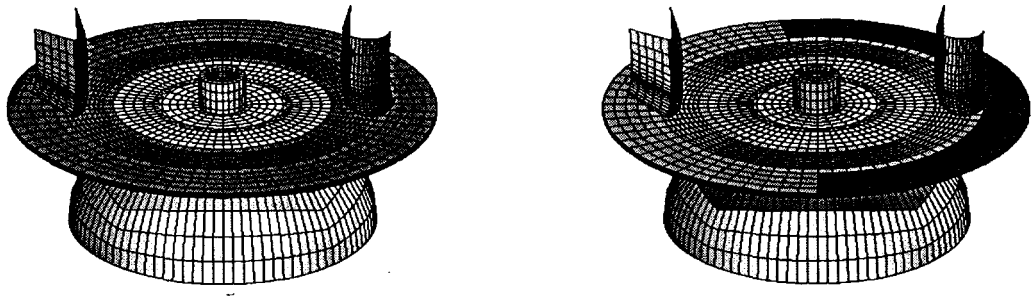


Figure 4. Four and eight-processor partitions for a spin forming problem. The model paritioned for four processors divides the plate into three concentric rings. The eight-processor partition leaves the middle ring uncut.

2.4.2 Parallel Automatic Contact

For arbitrary contact defined with sliding interface types 12-14, the analyst avoids the very time consuming task of defining the contact surfaces in the model. However, the parallel algorithm for these interface definitions is more expensive for two reasons. First, the search for contact is an expensive step, both on single processor computers and on parallel computers. Furthermore, for problems with considerable surface motion, the parallel versions of these algorithms accrue additional communication costs because the surfaces must be redistributed periodically among the processors. This surface redistribution load balances the contact calculations.

The efficiency of the parallel automatic contact can be improved by limiting the search domains with the DYNA3D keyword input options. Boxes may be used to delimit the search domains. Boxes save the computer time and storage required for the bucket sorts on the full mesh. This can save significant amounts of time for the large meshes designed for parallel computers. A further efficiency improvement is achieved with boxes by reducing the number of materials searched in each box using the material inclusion/exclusion options.

The parallel automatic contact algorithm is currently under very extensive development and optimization. The largest effort underway now is to load balance the parallel automatic contact. Until this effort and several other new techniques to optimize the algorithm are more complete, it is recommended that data sets using automatic contact definitions be tested for performance before use in production simulations. ParaDyn developers would be grateful for any benchmark tests that demonstrate a need for improved optimization when using the automatic contact algorithm. In return we will provide insights on modeling with the algorithms that may provide better performance. We will also use the benchmarks to guide our algorithm development for optimizing the method, particularly for models with rapidly changing surface topologies, nested surfaces, and meshes with moving objects.

2.4.3 Concluding Remarks on Parallel Contact

For a complex mesh it may be beneficial to use both local and arbitrary contact algorithms and to provide multiple instances for each type of contact. An obvious advantage in doing this is that the regions on a mesh without any contact are not included in either the partitioning for local contact or the search domains for arbitrary contact.

For problems with a relatively small amount of interface surfaces compared to the volume of materials and with localized interfaces (types 1-3 and 5-10), the balance numbers provided by the partitioning software can be used for selecting the number of processors to use. Furthermore, the parallel efficiency may be determined by the size of the largest contact surface.

For efficient parallel arbitrary contact (automatic contact) it is very important to use DYNA3D input options to limit the search domains. This algorithm is currently undergoing extensive code development for parallel optimization.

2.5 Boundary Conditions and Constraints

Parallel versions of boundary conditions and constraints are treated both in the partitioning software as well as in ParaDyn. Because the partitioning software uses special processing on selected boundary conditions and constraints, it is important to know which boundary conditions and constraints are treated with partitioning and how this affects the overall partitioning.

The following DYNA3D options are treated with the partitioning software [4,10]:

Symmetry planes with failure
Nodal forces and follower forces
Nodal constraints
Tie-breaking shell slidelines
Tied node sets with failure
Rigid body joints
Shell-solid interfaces
Discrete springs, dampers and masses
One-dimensional slidelines.

These DYNA3D objects contain nodes and elements that must be assigned to a single processor rather than divided across more than one processor. Nodes that need to be kept together are assigned to Special Nodal Points (SNP) sets in the partitioning software. Associated with each of the SNP sets is a Special Element (SE) set. The SE set consists of all elements that contain one or more nodes in the corresponding SNP set. Each SNP set and its associated SE set must be fully contained in a processor. As a result, large SNP or SE sets can constrain a mesh partitioning and limit the number of processors that can be used for the problem. Reference [4] and the online version provide examples.

3.0 ANALYSIS WITH PARADYN

An important consideration in our parallel algorithm development has been the requirement to provide automated tools for developing models for parallel computers. Automated tools have been developed for the new steps needed in either preprocessing the mesh or post-processing the binary plot databases.This chapter describes the ParaDyn software and how to use it.

3.1 The ParaDyn Software Set

The steps for running a ParaDyn simulation are shown in Figure 5.

Step 1. Mesh Generation		
Generate the mesh with special attention given to contact.		
Step 2. Mesh Partitioning		
Determine the optimal number of processors to use.		
Step 3. ParaDynAnalysis		
Run the ParaDyn analysis.		
Step 4. Visualization		
Display results on a parallel computer.	**OR**	Display results on a workstation.

Figure 5. Steps in preparing and running ParaDyn simulations.

The new preprocessing step for mesh partitioning is automated with script files, DYNAPART and DYNAPARTAGAIN. These files and other new utilities for parallel simulations are summarized in Figure 6.

The ParaDyn program is usually run with a utility, such as MPIRUN or POE, that copies the executable to the processors on a parallel computer. Most large simulations are run using batch processing software supplied by the Computer Center. Script files are often made available in public file systems to automate batch runs.

There is a new step in post-processing of the binary databases with either state data or time history data. This step involves combining the plot database families generated on each processor into a single plot database family that can be viewed with GRIZ4 or THUG. The two utilities, XMILICS and COMBINETHS, are used for combining the state and time history plot databases, respectively. GRIZ4 and THUG can be used to visualize the results on either a single node on the parallel computer or on a workstation. Often for larger problems, the combined databases are transported with FTP to a workstation for visualization.

The software products used for ParaDyn simulations are summarized in Figure 6. The next sections describe these products in detail. The final section summarizes the steps for using ParaDyn software and can be used as a handy reference once the details of each step are understood.

Task	Software	Input	Output
Mesh Partitioning	DYNAPART DYNAPARTAGAIN	DYNA3D input	Partition map, plot file for the partitioned mesh
Parallel DYNA3D	ParaDyn	DYNA3D input Partition map	Families of files: plot/time history output, restart, ...
File combiners: state and time history data	XMILICS COMBINETHS	Families by processor of state or time history data	One family for state or time history data
Visualization	GRIZ4, THUG	State, time history databases	Screen display, RGB output, ...

Figure 6. ParaDyn Software Products.

3.2 PATH Variable for ParaDyn

The directory containing ParaDyn software must be included in the list in the PATH variable. The following SET PATH command added to the .cshrc file places the directory containing ParaDyn software products into the path variable.

set path = (/usr/apps/bin/mdg $path)

3.3 Partitioning a Model

Before running ParaDyn, the model (including mesh, contact surfaces, constraints, and boundary conditions) must be partitioned using DYNAPART. It is often desirable to partition the model several times in order to select the number of processors to use for the model. The partitioning software provides statistics for guiding this selection process. DYNAPART and a related tool, DYNAPARTAGAIN, are script files which run several utilities for mesh and contact surface partitioning. The statistical output is written into the logfile generated when running these script files. Two utilties run within the scripts, METIS and PFGEN, produce the statistics for the partitioning of the model.

Since the partitioning software generates a number of intermediate files in the current working directory, running the partitioning scripts from a subdirectory is recommended. This conveniently groups the files containing the results generated for each mesh that is partitioned. The DYNAPART script is used the first time a mesh is partitioned. If the files from the first partitioning of a mesh are not destroyed, DYNAPARTAGAIN can be used for subsequent partitioning of the same mesh for a different number of processors. DYNAPARTAGAIN uses results saved from the DYNAPART execution and consequently, can skip over some of the time consuming steps in the DYNAPART script. It is very advisable to use DYNAPARTAGAIN when repartitioning a large model.

As a rule of thumb, an initial estimate for the optimal number of processors to use is roughly 2000 elements per processor. This estimate may be inadequate if contact surfaces, boundary conditions, or nodal constraints are restricting the partitioning (See discussions in sections 2.4 and 2.5.)

The execute lines for DYNAPART and DYNAPARTAGAIN are identical. The first argument is the name of the DYNA3D input file and the second argument is the number of processors requested for the mesh partitioning.

dynapart *infile np*
dynapartagain *infile np*

The output from the mesh partitioning is a file that is referred to as the *partition* file. The name of the partition file is *infile.np*. For the *d3sampl* input file for DYNA3D, the file name for a four-processor partition is *d3sampl.4*. The first several lines in the partition file contains partitioning statistics for the problem and the remainder of the file contains the list of element and node assignments to each processor. This file can be viewed with a text editor.

The partitioning scripts also generate a plot database that can be used to visualize the subdomains of the partitioned model. The root name for the family of files is *parplt*. GRIZ4 can be used to visualize this database. The result is a coloring of the mesh subdomains by processor and a corresponding map of color values associated with the processors.

Caution: The database family, *parplt*, is overwritten when the partitioning scripts are run again in the same directory. The database can be saved by renaming it before running the next partitioning of the model.

The parallel efficiency for a particular mesh partitioning can be evaluated from statistics provided on the screen or in a log file generated by the partitioning scripts. The METIS software calculates and prints statistics for evaluating the quality of the partitioning. This output is listed as the `Balance` number from METIS. The best balance is obtained when this number is as close as possible to a value of 1.000. Other statistics indicating the quality of the partitioning are printed by another tool, PFGEN, which is run by the scripts. The PFGEN output lists statistics about the shared-node communication resulting from the partition. The `Uniformity` number output by PFGEN measures the spread and the quantity of the communication among the processors. Again the best value for the communication uniformity is a number close to 1.0. Figure 7 is an excerpt from the screen output containing these statistics.

The balance and uniformity numbers do not provide a measure of the relative amount of communication occuring within any given processor. These statistics can be obtained by inspecting the partition file. The total number of nodes in a processor and the number of shared nodes in a processor are listed in the first lines of the partition file. See Figure 8. For an efficient calculation it is useful to keep the ratio of the number of shared nodes to the total number of nodes as small as possible. Ideally this ratio would be ten percent or less.

```
                Metis output statistics

Recursive Partitioning... --------------------------------
  Edge-Cuts:
    2-way   4-way   8-way  16-way
     5872   16500   46581   78503
  Balance: 1.002

                PFGEN output statistics

--  Starting to generate the partition-assignment file.

                                  Number of      Number of
                                    Node         Adjacent
                                Communications   Processors

        Average per processor:       875.0          3.1
         Max on any processor:      1520            5
       Total on all processors:     14000           50

  Uniformity: (max / average): 1.737          1.600
```

Figure 7. An excerpt from the screen containing Metis and PFGEN output statistics.

It may be convenient to save the screen output from the scripts into a file while also viewing the results. The saved log file can be used to inspect the statistics as well as error messages generated when the DYNAPART scripts are run. The UNIX TEE command can be used to do this as follows:

dynapart *infile np* |& **tee** *infile.log*

The details of the nodal communication (rather than averages) are listed in the first few lines of the partition file, *infile.np*. These can be conveniently examined with the UNIX HEAD command.

head –20 *infile.np*

The results of this command are shown in Figure 9.

Example: Finding an optimal number of processors

This example illustrates the use of the mesh partitioning tools to find the optimal number of processors to use for the first example problem included with the DYNA3D program.

Step 1. Set up a directory and partition the problem for two-processors.

The name of the input file is *d3samp1*. The directory for the partitioning is named *partition.*

```
mkdir partition
cd partition
ln -s ../d3samp1 d3samp1
```

The UNIX utility LN links the *d3samp1* file in a different directory from the current directory (*partition*) without making a copy. This saves disk space when when working with large files. It is also convenient to use LN when making runs using the same input file and varying the number of processors. This is the case when studying the scaling and efficiency for a model.

Execute DYNAPART the first time the mesh is partitioned.

```
dynapart d3samp1 2
```

The output files from this partitioning are *d3samp1.2* and the plot database, *parplt*. Next, visualize the partitioned mesh with GRIZ4.

```
griz4s –i parplt
```

Step 2. Look at the statistics from the DYNAPART interactive session.

The statistics from PMETIS and PFGEN are shown in the excerpt from the log file in Figure 9.

```
Running PMETIS
************************************************************
 METIS 2.0   Copyright 1995, Regents of the University of Minnesota

Graph Information ---------------------------------------------
   Graph: d3samp1.grfall, Size:   972,   19062, Parts:   2, Cto: 100
 Options: SHEM, BGKLR, GGPKL, Rec-Partition

Recursive Partitioning... --------------------------------------
  Edge-Cuts:
    2-way
      360
  Balance: 1.000

Running PFGEN

 --  Starting to generate the partition-assignment file.

                                 Number of      Number of
                                   Node          Adjacent
                                Communications   Processors

        Average per processor:        37.0           1.0
         Max on any processor:        37             1
       Total on all processors:       74             2

    Uniformity: (max / average):     1.000          1.000

 ++  Finished generating the partition-assignment file.
```

Figure 8. Partitioning statistics for a two-processor partition of the example mesh

Both of these statistics, the Metis Balance number and the Uniformity number written by PFGEN, suggest that the problem may be efficient if more processors are used for the partitioning of the model.

The following are the detailed statistics for the shared nodal data obtained by running the HEAD command on the partition file, *d3samp1.2*. Notice that only 37 if the 486 nodes in each processor are shared.

Detailed statistics from the partition file, d3samp1.2

```
 bar impact problem (gm cm microsec) INGDY1.dat                        97 large
#   NNPS   NHEXS  NBEAMS NSHELLS NTSHELLS NPROCS
    1369     972       0       0       0       2
# Number of Nodal Points per Processor: Processors 0 to    1
     703     703
# Number of Hexagonal Elements per Processor: Processors 0 to    1
     486     486
# Total number of Shared Nodal Points per Processor: Processors 0 to    1
      37      37
# Number of Adjacent Processors per Processor:Processors 0 to    1
       1       1
```

Figure 9. Total number of nodes, elements and shared nodes by processor for the two-processor partition

Step 3. Use DYNAPARTAGAIN to partition the model for four processors.

dynapartagain d3samp1 4

The partition file for this step is *d3samp1.4* and the plot database for the partitioned mesh overwrites the plot database for the previous partitioning.

The balance number from METIS for this partition is 1.000 indicating an even distribution of the computing work and communication among the four processors. The uniformity statistic from PFGEN shows that some processors have more communication. An inspection of the partition file, *d3samp1.4*, shows two of the processors engage in twice as much communication as the other two. The communicated nodes represent roughly 10% and 20% of the total number of nodes in the individual processors. These results are shown in Figure 10.

```
                    Partitioning d3samp1 for 4 processors

Running PFGEN
--  Starting to generate the partition-assignment file.

                                    Number of       Number of
                                      Node          Adjacent
                                  Communications    Processors

        Average per processor:        55.5            1.5
         Max on any processor:        74              2
       Total on all processors:      222              6

    Uniformity (max / average):     1.333           1.333

 ++  Finished generating the partition-assignment file.

           Detailed statistics from the partition file, d3samp1.4

 bar impact problem (gm cm microsec) INGDY1.dat                        88 large
#    NNPS    NHEXS  NBEAMS NSHELLS NTSHELLS NPROCS
     1369     972       0       0       0       4
# Number of Nodal Points per Processor:  Processors 0 to    3
     370     370     370     370
# Number of Hexagonal Elements per Processor:  Processors 0 to    3
     243     243     243     243
# Total number of Shared Nodal Points per Processor:  Processors 0 to    3
      37      74      37      74
# Number of Adjacent Processors per Processor:  Processors 0 to    3
    1       2       1       2
```

Figure 10. Total number of nodes, elements and shared nodes by processor for the four-processor partition

Step 4. Use DYNAPARTAGAIN to partition the mesh for eight processors.

This partitioning is most likely to be poor based on the four-processor partition. The results shown below verify this.

dynapartagain d3samp1 8

The METIS balance statistic is 1.004 when the model is partitioned for 8 processors. Nevertheless, an inspection of the partition file shows that the number of shared nodes per processor varies between 22% and 40% of the total number of nodes in each processor. This is an unacceptably high amount of communication. The statistics in the partition file are shown below.

We can conclude from the partitioning for two, four and eight processors that the optimal number of processors to use for this problem is four.

```
Detailed statistics from the partition file, d3samp1.8

bar impact problem (gm cm microsec) INGDY1.dat
88 large
#    NNPS    NHEXS   NBEAMS NSHELLS NTSHELLS NPROCS
     1369      972        0       0        0       8
# Number of Nodal Points per Processor:  Processors 0 to    7
      209      206      208      208      209      207      209      207
# Number of Hexagonal Elements per Processor:  Processors 0 to    7
      122      121      122      121      122      121      122      121
# Total number of Shared Nodal Points per Processor:  Processors 0 to    7
       82       82       83       46       46       83       83       83
# Number of Adjacent Processors per Processor: Processors 0 to    7
        2        2        2        1        1        2        2        2
```

Figure 11. Total number of nodes, elements and shared nodes by processor for the eight-processor partition

3.4 File Name Sequences

The names of restart files and plot databases generated in a ParaDyn run are different from those for a DYNA3D simulation.The names are lengthened to include a processor number in the string for the root name for the database. More generally, each processor generates a set of files equivalent to a DYNA3D run on a single processor. The files are distinguished from the standard set in a DYNA3D run by a shortening of the root name to three characters and the inclusion of a processor number in the string for the name. For a problem running on 1000 or more processors, the length of the string appended to the root name is adjusted to accommodate the number of digits contained

in the number of processors used. For example, for 1024 processors, four digits are added to the root name. Similarly for 10240 processors, five digits are added to the root name.The plot database names shown in Figure 12 are the default names generated for MILI state database files for a problem with less than 1000 processors.

1. State and time history databases: Root names m_p and tht			
P_0	**P_1**	**P_2**	**P_3**
m_p000A m_p000	m_p001A m_p001	m_p002A m_p002	m_p003A m_p003
m_p00001	m_p00101	m_p00201	m_p00301
m_p00002	m_p00102	m_p00202	m_p00302
...	...	...	...
...	...	...	...
m_p00099	m_p00199	m_p00299	m_p00399
2. Restart database names: Root names dmp			
P_0	**P_1**	**P_2**	**P_3**
dmp000p01	dmp001p01	dmp002p01	dmp003p01
dmp000p02	dmp001p02	dmp002p02	dmp003p02
dmp000p03	dmp001p03	dmp002p03	dmp003p03
3. Text output file names			
P_0	**P_1**	**P_2**	**P_3**
d3hsp	d3hsp001	d3hsp002	d3hsp003
frc000	frc001	frc002	frc003

Figure 12. Naming convention for files generated during a ParaDyn simulation.

3.5 ParaDyn Command Lines

The partition file must be copied into a file named *partfile* before executing ParaDyn. Do not remove the original partition file. You can also use a soft link if the files are big. This is explained in the examples. The partition file is used with the post-processing software.

The ParaDyn execute line arguments are identical to the DYNA3D execute line. The restarts for ParaDyn likewise use the same execute line as those used in executing a DYNA3D restart. The command lines for a ParaDyn initial run and a restart execution are illustrated in the following two examples.

paradyn i=*infile* **l=***filelength*
paradyn i=*restart* **r=***dumpfile* **l=***filelength*

paradyn i=*infile* **q=***nseconds*
paradyn c=*lastdump* **q=***nseconds*

The first set of execute lines uses the standard input lines for restarts described in the DYNA3D manual. The second set of execute lines illustrates a restart method which starts from the last successfully written restart file.

Caution: The standard output files, (d3hsp, d3hsp0001, ...) are overwritten when a subsequent run is made in the same directory. It is generally a good practice to at least save the file *d3hsp* into a different name between restarts of the problem. This output file, *d3hsp*, usually has results that are of value at the end of a long sequence of restarts. For instance, filenames, input options, and dynamic relaxation results are written into the file *d3hsp*.

The **c=** option specifies the name of a file which is always *lastdump*. ParaDyn generated the *lastdump* file at the end of the most recent run made in the directory. ParaDyn writes one line in the *lastdump* file. This line conains the name of the dump restart file which has the last successfully written state data. This is a very important option to use on some parallel computers, particularly the ASCI Blue computer at LLNL.

Notice that it is not necessary to specify an input file for a restart execution. If the input file is not specified, all input options will remain the same as those specified in the preceding run. These values from the previous run are stored in the restart dump file and read in during the input phase of the restart run.

The **q** option provides a capability for terminating a problem on the ASCI Blue Pacific system. The value, in seconds, specifies the number of seconds (by the wall clock) to run the ParaDyn simulation before ParaDyn stops itself with a normal termination. This option is often used to allow

the analyst to specify a wall-clock termination time slightly shorter than the batch time he selects. The extra time allows for final file writes and close operations for the database families and other output files.

The **l=** option provides a method for increasing the maximum file length for each member of a database family of plot files. The size specified is in units of Megabytes. Caution: If the file length option is used on the initial ParaDyn run, it must be used on all subsequent restarts.

Example: Execute lines for ParaDyn
Consider an initial ParaDyn run using the DYNA3D input file, *d3samp1* and a restart run using an input data file, *irestart*. Following the naming convention in Figure 12, the initial run and restart run are executed with the following ParaDyn input arguments.

paradyn i=d3samp1 l=10
paradyn i=irestart r=dmp000p01 l=10

Notice, the file lenth is specified on both the initial run and on the restart.

This next example illustrates the use of the **q=** termination option.

paradyn i=d3samp1 q=3000 l=100
paradyn c=lastdump q=3000 l=100

This problem will be submitted to batch to run for one hour (3600 seconds) and ten minutes before the end of the run (afer 3000 seconds) ParaDyn will call the termination routine to close files and exit. The maximum length of the files in the plot databases is set to 100 Megabytes. The name of the restart file is the name listed on the first line of the file named *lastdump*.

3.5.1 ParaDyn Run Interactively

ParaDyn can be run interactively using the POE utility on an ASCI Blue computer at LLNL. This system has four processors per node and five hundred or more nodes. The POE utility runs ParaDyn with an input specifying the number of nodes and optionally, the total number of processors. The options for specifying the number of nodes and processors are inserted between the ParaDyn command and the execute line options for ParaDyn. If only one processor per node is being used, then it is sufficent to simply specify the number of nodes as follows:

poe paradyn -nodes *numnodes executeline*

More than one processor per node can be used by specifying the number of nodes with **-nodes** and the total number of processors with **-procs**.

poe paradyn -nodes *numnodes* **-procs** *np executeline*

Example: Interactive ParaDyn run on the ASCI Blue Pacific
The following are command lines for executing ParaDyn with the DYNA3D test case 1 as an input file and using either four or five processors.

poe paradyn -nodes 4 -procs 1 i=d3samp1
poe paradyn -nodes 1 -procs 4 i=d3samp1 4
poe paradyn -nodes 2 -procs 5 i=d3samp1 4

The first example above will run using one processor on each of four nodes. The second example will run with four processors on one node. The third example will run on two nodes using five of the eight processors.

ParaDyn can be run interactively using the MPIRUN utility on an Origin 2000 computer. The MPIRUN utility runs ParaDyn for *np* processors as follows.

mpirun –np *np* **paradyn** *dyna3d execute line*

Example: Interactive ParaDyn run on the Origin 2000
The following is the command line for executing ParaDyn with the DYNA3D test case 1 as an input file and using 4 processors.

mpirun –np 4 paradyn i=d3samp1

3.5.2 ParaDyn Run with Batch

ParaDyn production simulations are run under the DPCS batch processing system on the ASCI Blue computers at LLNL. To set up a problem for batch, a script file is prepared and then submitted to the batch system with the PSUB utility. Lines included in a typical script file for a ParaDyn simulation are shown in Figure 13.

The script file is submitted to batch for processing using the PSUB utility.

psub *scriptfile*

Use the PSTAT utility for interrogating the status of runs submitted with PSUB. A status of RUN indicates the job is running on the parallel computer. Type SPJSTAT to further check on the status of the run after PSTAT indicates it is in a RUN state.The online MAN pages can be viewed to study other options provided by the PSUB and PSTAT utilities.

```
# PSUB -eo
# PSUB -tH 2:00          # Job is to run for a maximum of 2 hours
# PSUB -ln 8
# PSUB -g 32
printenv
echo "started at"
date
cd /p/gk2/loginname/dynatests
# Allow 10 minutes (600 seconds) for file cleanup
poe paradyn i=dynin q=6600 l=100
echo "ended at"
date
```

Figure 13. A typical batch script file for a ParaDyn simulation

3.6 Visualizing ParaDyn Results

There are two choices for post-processing the state database output from a ParaDyn analysis. First the database families from the processors must be combined with the utility XMILICS. Then the results may be viewed on the parallel computer with GRIZ4. Alternatively, the combined database families may be transported to a workstation and viewed with GRIZ4.

There are some tradeoffs to be made in selecting either of these choices. For large problems particularly, the rendering and the display of a frame sent over a network to a workstation can be as long as twenty or thirty minutes. In this case better interactivity is possible by using FTP to send the combined database over to a local workstation. The additional resources needed for this improved interactivity are twice the disk space on the parallel computer, a workstation with high-speed graphics capabilities, and enough disk space on the workstation for the full database. Even with this additional hardware, the viewing of results is delayed by the time it takes to FTP the databases to the workstation. In the future, parallel rendering and visualization tools will become available that will eliminate some of these delays.

3.6.1 MILI Database Files

MILI plot databases are new. The format for the MILI databases provide a considerable amount of flexibility for designing material templates that display results specific to each material type as well as the flexibility for selecting which state data fields should be included in a database generated by a parallel analysis. Equally as important, the menus in GRIZ4 will reflect those data fields, result fields, and names which have been written into the MILI files. GRIZ4 must be used for displaying MILI plot databases.

Recent code development in both ParaDyn and DYNA3D require the use of the MILI database format. In addition, the MILI databases have been extensively tested in large benchmarks and in production runs at both Livermore and Los Alamos. For these reasons, we encourage their use. The old TAURUS database format used with ParaDyn is described in Section 4.5.

The use of GRIZ4 is also strongly recommended because of its expanded capabilities to handle the menu- generation for MILI databases. In the future, Mili database formats will be developed to provide many new features for material models, to display new results (not currently in TAURUS databases), to highlight boundary conditions, and to provide flexibility for limiting the size of parallel databases.

The MILI database software was released with ParaDyn version 1.01 and will be the default database type in future versions of ParaDyn. In addition, it is possible to select one or both database formats in the keyword control options input section of the input file. The keyword specification is:

mili_plot *miliflag*
endfree

The value of the flag is set to 1 to select the MILI database format and is set to 0 to deselect the database type.

Example: Specify MILI for the plot database format in a ParaDyn analysis.
Add the following keyword input to the keyword control options section of the input file.

mili_plot 1
endfree

3.6.2 Combining Parallel Databases

The utility XMILICS combines the state database from a ParaDyn run. The combined databases can then be viewed with GRIZ4. The execute line for XMILICS is

xmilics -i *infileroot* **-o** *outfileroot* **-c** *partfile*

The first argument is the root name for the families of state databases and the second argument is the root name for the output files with the combined databases. The third argument is the name of the partition file.

The utility COMBINETHS combines the time history database from a ParaDyn run. The combined time history database can then be viewed with the THUG uility. The execute line for COMBINETHS is

combineths *infileroot outfileroot partfile* **0 0**

The first argument is the root name for the families of time history databases and the second argument is the root name for the output files with the combined databases. The last two arguments must both be 0.

Example: XMILICS and COMBINETHS execute lines

xmilics -i m_p -o mout -c partfile
combineths plt oplt partfile 0 0

See Figure 14 for the list of output files generated with each combining utility.

```
            XMILICS output files
moutA, mout00, mout01, mout02, mout03,…..,

            COMBINETHS output files
otht00, otht01, otht02,….
```

Figure 14. The names of files in output databases from XMILICS and COMBINETHS.

On the workstation or the parallel computer, these databases are viewed with the GRIZ4 and THUG utilities.

griz4s –i mout
thug –i otht

3.7 Summarized Steps for Using ParaDyn

Step 1. Set the PATH variable for the ParaDyn software.

This line is added in the .cshrc file the first time the software is used.

set path=(/usr/apps/bin/mdg $path)

Step 2. Partition the mesh.

Make a directory for partitioning. Find the optimal number of processors for the run.

mkdir partition
cd partition
ln -s ../d3samp1 d3samp1
dynapart d3samp1 2
dynapart d3samp1 4
dynapart d3samp1 8
mv d3samp1.2 d3samp1.4 ../
cd ..

Step 3. Execute ParaDyn initial and restart runs

Make a directory for the simulation. Link the input and partition files into the directory.

mkdir rundir

cd rundir
ln -s ../d3samp1 d3samp1
ln -s ../d3samp1.4 partfile
paradyn i=d3samp1

Save the *d3hsp* output file between restarts. Input new options with the file *irestart.*

mv d3hsp hsprun1
paradyn i=irestart d=dmp000p01
mv d3hsp hsprun2

Use the *lastdump* file to get the name of the last successfully written restart dump file. There are no resets of the input options for the next restart.

paradyn c=lastdump

Step 4. Visualize the results.

Combine the state database and the time history databases.

xmilics -i m_p -o mout -c partfile
combineths tht otht d3samp1.4 0 0

Move the combined state and time history databases to the workstation, if desired. View the results on either computer.

griz4s –i mout
thug –i otht

4.0 INPUT FOR PARALLEL SIMULATIONS

This chapter describes data input and special instructions used for parallel simulations only. Instructions for running multiple ParaDyn analyses and special instructions for post-processing results written to text files are described here.

4.1 Static Initialization and Dynamic Analysis

Dynamic relaxation and stress initialization techniques are described in the DYNA3D manual. The following example shows one method for initializing a problem with a load, generating the stresses from a relaxation procedure, and starting a time dependent simulation. Three runs are used for this procedure.

Step 1. Run the relaxation step first.
This feature is activated by a flag on the load curve definition cards in the standard DYNA3D input.

paradyn i=input1

This creates two dump files for each processor. For processor zero, they are *dmp000p01* and *dmp000p02*.

Step 2. This step generates a stress output file.
Restarting the previous run for one time step to generate the stress file.

paradyn i=restart r=dmp000p02

This run creates the stress file, *d3sppp*, where *ppp* is the processor number.

Step 3. This steps runs a dynamic analysis using the stress file from the static initialization.

paradyn i=input2 n=d3s

4.2 The NIKE–DYNA Link File

The NIKE-DYNA link file is often used to start up a new run using stress results from a previous run on the same mesh. The details for this technique are described in section 2.19 of the DYNA3D manual. The first run to generate the stress file must include the following free format input to generate the output database with the stresses. The root name for the link file is specified on the following keyword input line.

nikefile *stressfile*
endfree

A ParaDyn execution with this input selecting the name, *str*, for the NIKE-DYNA link file will create database families for each processor labeled

str001, str002, ... strppp, ...

where *ppp* is the processor number.

A second run uses the same input mesh but other parameters in the input, such as load curves, may be changed. (Although the control options can be changed, it is not a requirement, and the same data input file can be used for both runs.) The second run reads the stress databases, stores the stress data, and uses the deformed nodal coordinate values from the first run. Other parameters associated with the original run may also be set and are described in the section **Keyword-Based Control Features** of the DYNA3D manual.

Example: Generate a NIKE-DYNA link file with ParaDyn

Consider a run with an input file, *d3st*, specifying the name stress on the **nikefile** keyword-input line. A second run will use the generated stress database by specifying the **m=** option on the execute line and new DYNA3D options in the input file, *d3st1*.

paradyn i=d3st
paradyn i=d3st1 m=stress g=glt

In the second run, the plot state database file family is given a new root name, *glt*.

4.3 Multiple Versions of Running Restart Files

The following describes a new capability for generating multiple versions of a running restart dump file. The option is selected with a keyword input and optionally specifying the running restart file names on the ParaDyn execute line. The keyword input is

numrrf *nvers*

The integer value, *nvers*, specifies the number of different versions of the running restart file to save on disk. The running restart family names are incremented through a set of family members until nvers of them have been written. Once this limit is reached the next running restart over writes the first running restart file in the set. The examples below illustrate the cycling through of the names of the versions of the running restart files.

The number of cycles between the running restart dump files is selected as usual in the fifth data field, columns 36-40, of DYNA3D control card 6. The default names for the running restart file for ParaDyn are *rsfnnnmm*, where *nnn* is the three-digit processor number and *mm* is the family-member number. To select the name of the running restart file, use the **a=** option on the ParaDyn execute line.

Example: Specify multiple versions of the running restart file

Set up the input file to save three different versions of the running restart file.

numrrf 3
endfree

Suppose a problem is run for 100 cycles and the number of cycles between running restart dumps is 30. The files generated for a 4-processor run will be:

rsf00001 rsf00101 rsf00201 rsf00301 (cycles 30)
rsf00002 rsf00102 rsf00202 rsf00302 (cycles 60)
rsf00003 rsf00103 rsf00203 rsf00303 (cycles 90)

Suppose this problem is run for another 150 cycles, then the running restart files will be

rsf00002 rsf00102 rsf00202 rsf00302 (cycle 150)
rsf00001 rsf00101 rsf00201 rsf00301 (cycle 120)
rsf00003 rsf00103 rsf00203 rsf00303 (cycle 180)
rsf00001 rsf00101 rsf00201 rsf00301 (cycle 210)
rsf00002 rsf00102 rsf00202 rsf00302 (cycle 240)

In the above, the cycle (30, 60, 90) files were over written by the cycle (120, 150, 180) files, respectively. And finally, the cycle (210, 240) files over write the cycle (120, 150) files.

Example: Select a new name for the running restart files
Suppose instead in the previous example that the initial problem is restarted from cycle 60 and runs up to cycle 250. A new name can be selected for the versions of the running restart files as follows:

paradyn i=inrest,r=rsf00002,a=nrf

The files generated will be

nrf00001 nrf00101 nrf00201 nrf00301 (cycle 90)
nrf00002 nrf00102 nrf00202 nrf00302 (cycle 120)
nrf00003 nrf00103 nrf00203 nrf00303 (cycle 150)
nrf00001 nrf00101 nrf00201 nrf00301 (cycle 180)
nrf00002 nrf00102 nrf00202 nrf00302 (cycle 210)
nrf00003 nrf00103 nrf00203 nrf00303 (cycle 240)

4.4 Nodal Force Output

The nodal force output is described in the keyword input and Section 4.47 of the DYNA3D input manual. Nodal force output is selected using the keyword input variable **nfrout**. The value of this keyword input variable is the total number of nodes that will be written into the output files.

Output format

For each time at which the nodal forces are desired, the following set of data are written:

Card 1 (E12.5,1x,i8)
Col. 1-12 Time at which forces have been computed E12.5

Col.14-20	Number of nodes (nfrout)	I8

Cards 2 to Card 1+nfrout (i8,3e12.7)

Col. 1-8	Node number	I8
Col. 9-20	Force in the x-direction	E12.5
Col. 21-32	Force in the y-direction	E12.5
Col. 33-44	Force in the z-direction	E12.5

The name of the text output file containing the nodal force data is *nodfrc* for a DYNA3D run.

ParaDyn generates two kinds of text output files. The first output file (generated by processor zero only) contains the times at which the forces are output for the selected nodes. The name of this file is *nfrctimes*. The remaining output data, the node numbers and three components of the force, are written to a set of files generated by all of the processors. The files are named nfrc*nnn*, where *nnn* is the three-digit processor number. By separating the data in this way, the following UNIX utilities can be used to combine the data into a single file, with the time steps for the output at the top, followed by the list of nodes and forces for each time step.

cat nfrc* | sort -n - > forceout

Example: Combine nodal force output from ParaDyn

The following is an example in which two nodes, (12,144), were selected and written at three different time steps. The nodes were merged using the files from the processors with the previous CAT and SORT utilites. The results are

```
0.00000E+00 2
0.44627E-06 2
0.89344E-06 2
          12  0.00000E+00 0.00000E+00 0.00000E+00
          12  0.00000E+00 0.00000E+00 0.00000E+00
          12  0.00000E+00 0.00000E+00 0.00000E+00
         144 0.00000E+00 0.00000E+00 -0.14000E-11
         144 0.00000E+00 0.00000E+00 -0.14000E-11
         144 0.00000E+00 0.00000E+00 -0.15000E-11
```

To eliminate the time stamp data at the beginning of the file (and read it from the file *nfrctimes* instead in a post-processor), use the following UNIX utilities for combining the output.

cat nfrc??? | sort -n - > forceout

4.5 TOPAZ3D Temperature Input

The TAURUS database with nodal temperature data generated by the TOPAZ3D program can be input to a ParaDyn simulation. Input options are completely compatible with the DYNA3D/ TOPAZ3D temperature input options. The restriction for using this feature is that the same mesh must be used for both programs.

In the future we expect to develop a MILI database to link TOPAZ3D and DYNA3D/ParaDyn. The utility XMILICS currently has the capability to both combine and split MILI databases into a specified number of processors. This will allow a link between a parallel or single processor TOPAZ3D simulation and a ParaDyn simulation where the number of processors used for each code is not restricted to be the same.

4.6 TAURUS Database Files

Many of the new ParaDyn code developments require the use of the flexibile file formats provided by the MILI I/O library. For this reason, TAURUS databases are no longer generated by default in ParaDyn. Keyword input may be used to select TAURUS databases for those special circumstances where they may be needed. An example may be a restart from an older run ParaDyn run. The file name sequences for TAURUS databases is shown in Figure 15.

TAURUS plot databases are selected in the keyword input section of the DYNA3D/ParaDyn input file. Both MILI and TAURUS databases can be output in the same run. This capability is usually used by the code developers and not recommended for long analysis runs. The following two keywords select and deselect the TAURUS and MILI databases.

mili_plot *miliflag*
taurus_plot taurusflag
endfree
The value of either flags is set to 1 to select the database type and is set to 0 to deselect the database type.

1. State and time history databases: Root names plt and tht			
P_0	P_1	P_2	P_3
plt000	plt001	plt002	plt003
plt00001	plt00101	plt00201	plt00301
plt00002	plt00102	plt00202	plt00302
...	...	...	...
...	...	...	...
plt00099	plt00199	plt00299	plt00399

Figure 15. Naming convention for TAURUS database file families generated during a ParaDyn simulation.

The PGRIZ utiltity is only available on the Livermore computer systems. This utility combines the TAURUS databases in memory when visualizing the plot files on the parallel computer. The execute line for the PGRIZ tool is

pgriz -i *filename* **-p** *partfile* **0 0**

The first argument, *filename*, is the full name of the first file in the database family for processor 0. The second argument, *partfile*, is the name of the partition file. The last two arguments are always set to zero.

Example: PGRIZ execute line

An example input line for the test case in the previous example is

pgriz -i plt –p d3samp1.4

The utilities XMILICS and COMBINETHS combine the Taurus state and time history databases from a ParaDyn run. The state databases can be viewed with GRIZ4 and the time history databases with THUG.

The XMILICS execute lines is

xmilics -i *infileroot* **-o** *outfileroot* **-c** *partfile*

The first and second arguments are the root names of the input TAURUS database and the combined output MILI database. The third argument is the name of the partition file.

The COMBINETHS execute line is

combineths *infileroot outfileroot partfile* **0 0**

The first argument is the root name for the input families of time history databases and the second argument is the root name for the output family of files with the combined databases. The last two arguments must both be 0.

Example: XMILICS and COMBINETHS execute lines

Consider the state and time history databases for the previous four-processor example. The root names for the plot databases and time history databases are plt and ths, respectively. The following execute lines will generate an output family of files with the root names oplt and oths, respectively.

xmilics -i m_p -o oplt -c d3samp1.4
combineths tht otht d3samp1.4 0 0

```
          XMILICS output files
moutA, mout00, mout01, mout03, ....

          COMBINETHS output files
otht00, otht01, otht02,....
```

Figure 16. The names of files in output databases from XMILICS and COMBINETHS.

On the workstation, these databases are viewed with the GRIZ4 and THUG utilities.

griz4s –i oplt00
thug –i otht

5.0 FUTURE ENHANCEMENTS

Parallel algorithm design is an integral part of the software development cycle for the DYNA3D/ ParaDyn software. The Design Phase includes the design of the approximate technique for the mechanics, the algorithm design for single processor computers, and the complementing design for parallel computers. Once a design is in place, frequently the software development proceeds with a serial (DYNA3D) implementation and testing. This then is followed by a parallel implementation and testing. The following features implemented in DYNA3D are future enhancements to the ParaDyn software.

1. Contact algorithms

LaGrange constraint forms of contact.

2. Rigid body algorithms

Deformable and rigid material switching algorithm.

3. Boundary conditions

Rigid nodal constraint sets.

4. New elements

Cohesive and delaminated elements.

There are two DYNA3D features which will not be implemented in parallel as follows:

1. Coupled codes

Madymo DYNA3D link capability.

2. Contact algorithms

Slide line type 11. This algorithm is an obsolete version of the SAND algorithm. The replacement algorithm in sliding interface type 14 is implemented in parallel.

REFERENCES

1. , Lin, Jerry I., "DYNA3D: A Nonlinear, Explicit, Dimensional Finite Element Code for Solid and Structural Mechanics—User Manual," Lawrence Livermore National Laboratory, Livermore, California, UCRL-MA-107254 Rev. 2, **1999**.
2. Speck, D. E., Dovey, D. J., "GRIZ: Finite Element Analysis Results Visualization for Unstructured Grids—User Manual," Lawrence Livermore National Laboratory, Livermore, California, UCRL-MA-115696, **2000**.
3. Speck, D. E., "MILI: A Mesh I/O Library—Programmer's Reference", Lawrence Livermore National Laboratory, Livermore, California, UCRL-MA (in publication) **2000**.
4. Sherwood, Robert J., "DynaPart Auxiliary Documentation Files", Lawrence Livermore National Laboratory, Livermore, California, UCRL-ID-137888, **2000**.
5. Schauer, D. A., Hoover, C.G., Kay, G. J., Lee, A.S., and De Groot, A.J., "Crashworthiness Simulations with DYNA3D", Paper No. 961249, Transportation Research Board, **1996**.
6. Karypis, G. and Kumar, V., "METIS 3.0: Unstructured Graph Partitioning and Sparse Matrix Ordering System," University of Minnesota, Department of Computer Science, **1997**. See also http://www-users.cs.umn.edu/~karypis/metis/main.shtml.
7. Karypis, G. and Kumar, V., "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," SIAM Journal on Scientific Computing, **1998**. A short version appears in Intl. Conf. on Parallel Processing, **1995**.
8. Karypis, G. and Kumar, V., "Multilevel k-way Partitioning Scheme for Irregular Graphs," Journal of Parallel and Distributed Computing, **1997**.
9. Hendrickson, B. and Leland, R., "An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations," Sandia National Laboratory Report Number SAND92-1460, **1992**.
10. Hoover, C.G., Badders, D. C., De Groot, A.J., and Sherwood, R. J., "Parallel Algorithm Research for Solid Mechanics Applications Using Finite Element Analysis," Lawrence Livermore National Laboratory, Livermore, California, UCRL-ID-129202, **1997**.